Unit 4

Pipeline and Vector Processing

PIPELINING AND VECTOR PROCESSING

- Parallel Processing
- Pipelining
- Arithmetic Pipeline
- Instruction Pipeline
- RISC Pipeline
- Vector Processing
- Array Processors

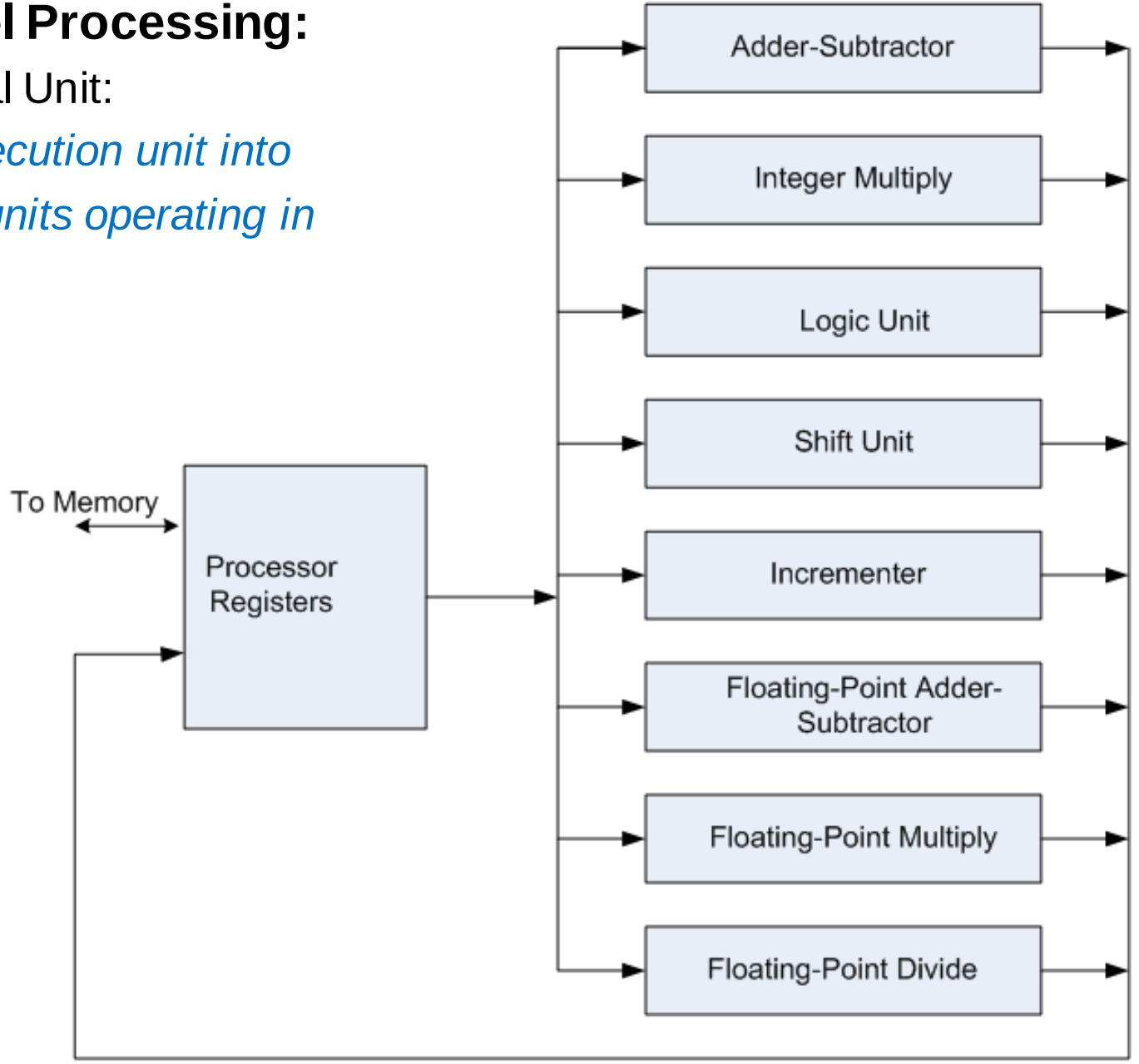
PARALLEL PROCESSING

- Parallel processing is a term used for a large class of techniques that are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of a computer system.

PARALLEL PROCESSING

- **Example of parallel Processing:**
  - Multiple Functional Unit:

    Separate the execution unit into eight functional units operating in parallel.



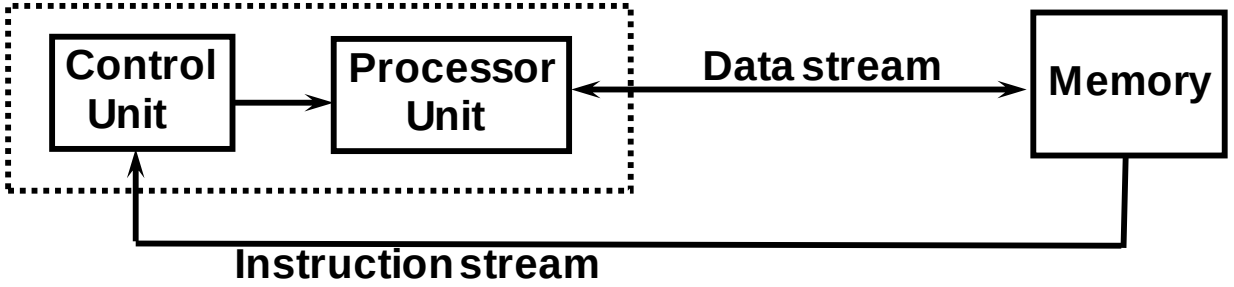
PARALLEL COMPUTERS

Architectural Classification

- **Flynn's classification**
  - » Based on the multiplicity of *Instruction Streams* and *Data Streams*
  - » Instruction Stream
    - Sequence of Instructions read from memory
  - » Data Stream
    - Operations performed on the data in the processor

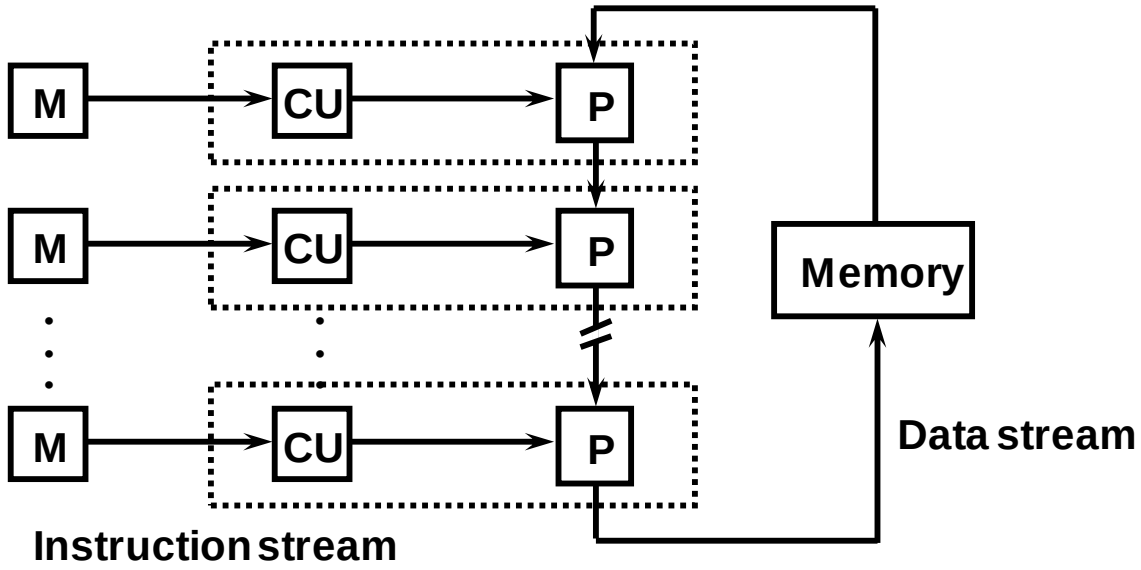
		Number of *Data Streams*	
		Single	Multiple
Number of *Instruction Streams*	Single	**SISD**	**SIMD**
	Multiple	**MISD**	**MIMD**

SISD COMPUTER SYSTEMS



- **Characteristics:**
  - ➢ One control unit, one processor unit, and one memory unit
  - ➢ Parallel processing may be achieved by means of:
    - ✓ multiple functional units
    - ✓ pipeline processing

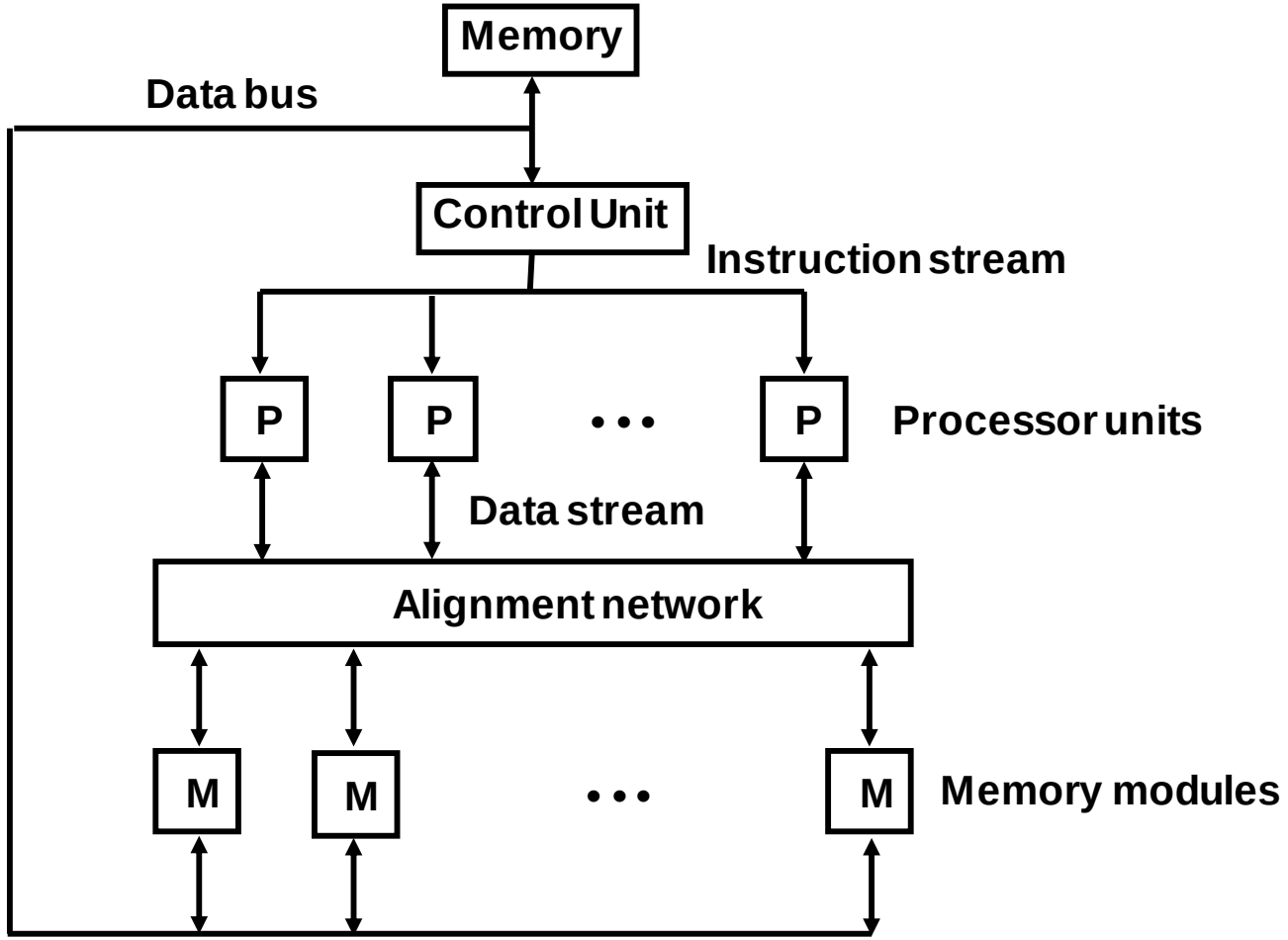
MISD COMPUTER SYSTEMS



Characteristics

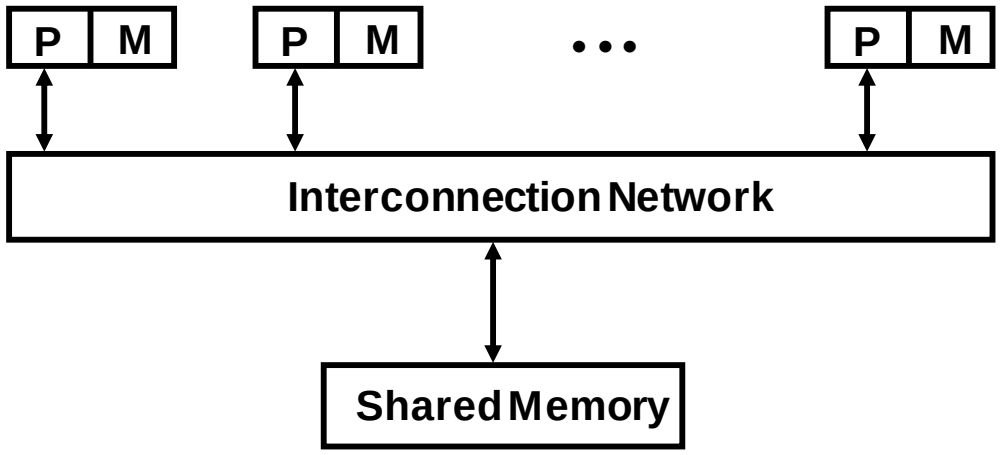
- There is no computer at present that can be classified as MISD

SIMD COMPUTER SYSTEMS



- **Characteristics**
  - Only one copy of the program exists
  - A single controller executes one instruction at a time

MIMD COMPUTER SYSTEMS



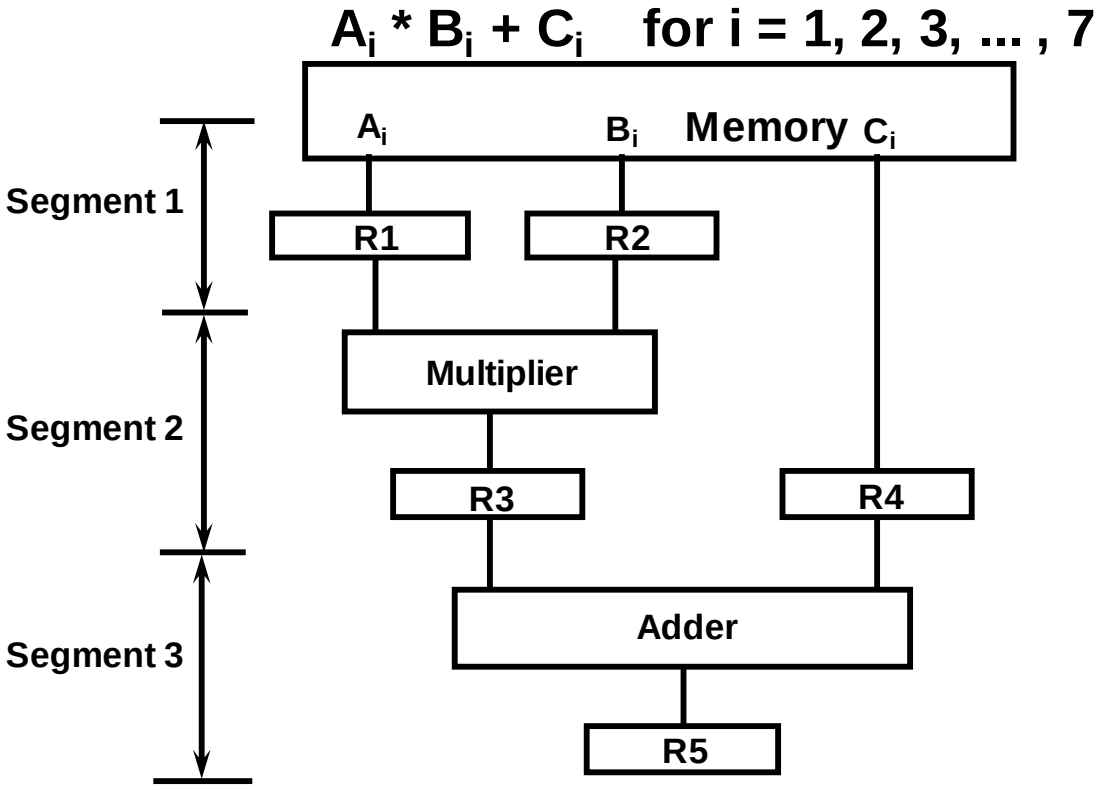
- **Characteristics:**
  - Multiple processing units (multiprocessor system)
  - Execution of multiple instructions on multiple data

- **Types of MIMD computer systems**
  - Shared memory multiprocessors
  - Message-passing multicomputers (multicomputer system)

- The main difference between multicomputer system and multiprocessor system is that the multiprocessor system is controlled by one operating system that provides interaction between processors and all the component of the system cooperate in the solution of a problem.

PIPELINING

- A technique of decomposing a sequential process into suboperations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments.



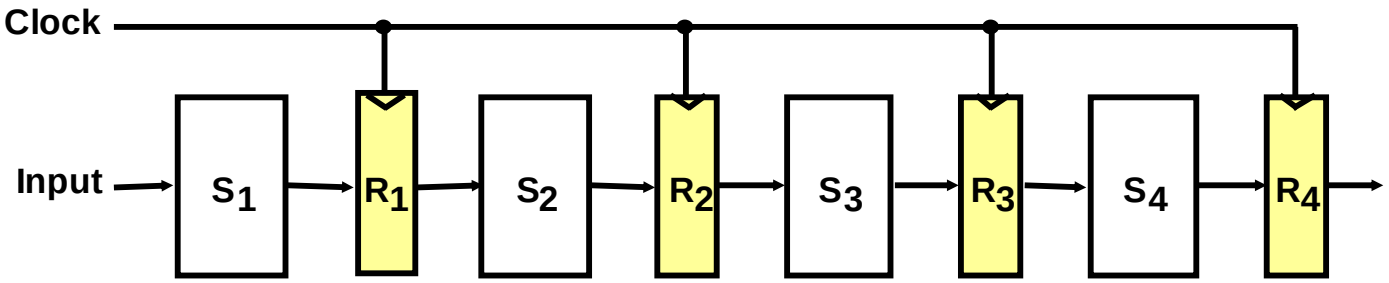
Suboperations in each segment:		
	$R1 \leftarrow A_i$, $R2 \leftarrow B_i$	Load A_i and B_i
	$R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$	Multiply and load C_i
	$R5 \leftarrow R3 + R4$	Add

OPERATIONS IN EACH PIPELINE STAGE

Clock Pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
1	A1	B1	---	---	-------
2	A2	B2	A1 * B1	C1	-------
3	A3	B3	A2 * B2	C2	A1 * B1 + C1
4	A4	B4	A3 * B3	C3	A2 * B2 + C2
5	A5	B5	A4 * B4	C4	A3 * B3 + C3
6	A6	B6	A5 * B5	C5	A4 * B4 + C4
7	A7	B7	A6 * B6	C6	A5 * B5 + C5
8			A7 * B7	C7	A6 * B6 + C6
9					A7 * B7 + C7

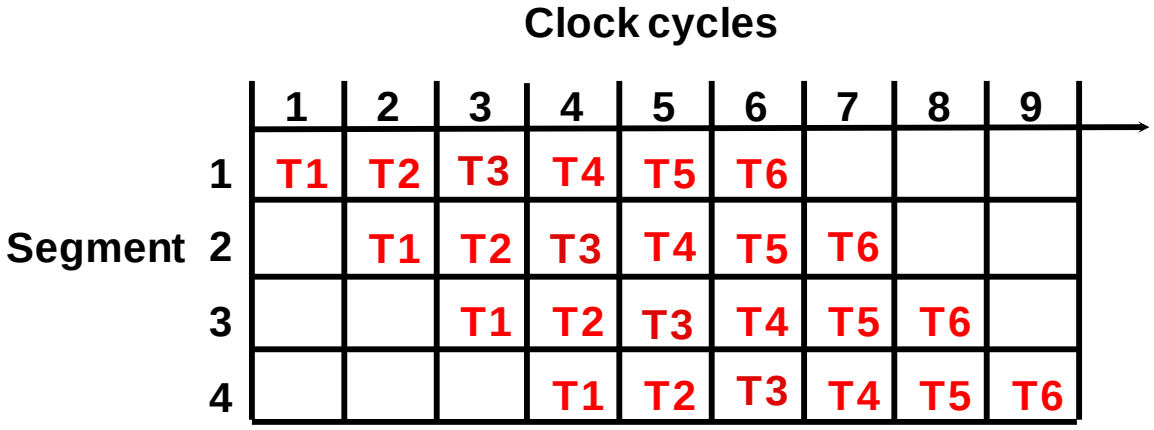
GENERAL PIPELINE

- **General Structure of a 4-Segment Pipeline**



- **Space-Time Diagram**

  The following diagram shows 6 tasks T1 through T6 executed in 4 segments.



No matter how many segments, once the pipeline is full, it takes only one clock period to obtain an output.

PIPELINE SPEEDUP

Consider the case where a k-segment pipeline used to execute n tasks.

- n = 6 in previous example
- k = 4 in previous example

- **Pipelined Machine (k stages, n tasks)**
  - The first task t1 requires k clock cycles to complete its operation since there are k segments
  - The remaining n-1 tasks require n-1 clock cycles
  - The n tasks clock cycles = k+(n-1) (9 in previous example)
- **Conventional Machine (Non-Pipelined)**
  - Cycles to complete each task in nonpipeline = k
  - For n tasks, n cycles required is = nk
- **Speedup (S)**
  - S = Nonpipeline time /Pipeline time
  - For n tasks: S = nk/(k+n-1)
  - As n becomes much larger than k-1; Therefore, S = nk/n = k

PIPELINE AND MULTIPLE FUNCTION UNITS

Example:

- 4-stage pipeline
- 100 tasks to be executed
- 1 task in non-pipelined system; 4 clock cycles

Pipelined System : $k + n - 1 = 4 + 99 = 103$ clock cycles

Non-Pipelined System : $n*k = 100 * 4 = 400$ clock cycles

Speedup : $S_k = 400 / 103 = 3.88$

Types of Pipelining

- Arithmetic Pipeline
- Instruction Pipeline

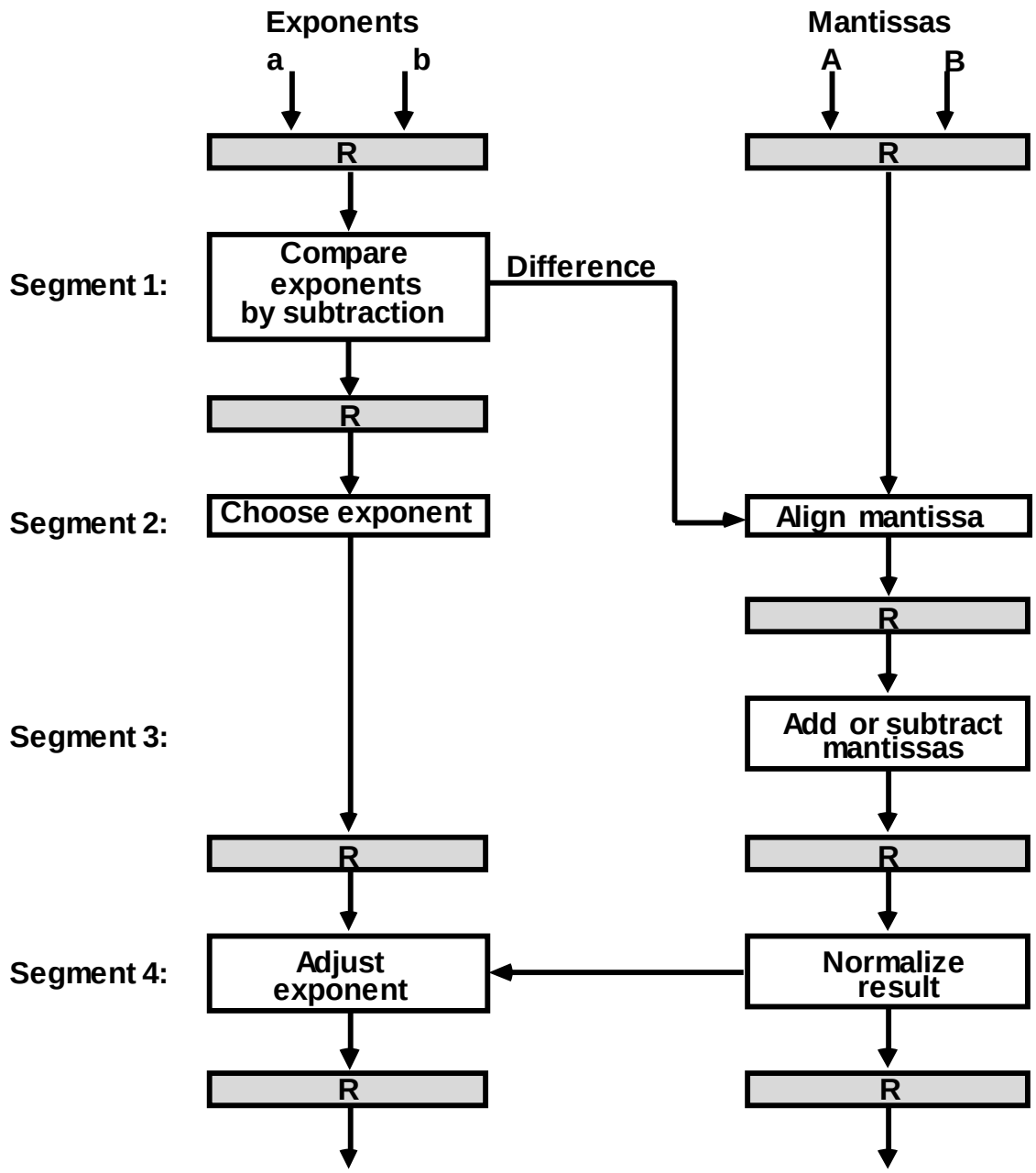
ARITHMETIC PIPELINE

Floating-point adder

[1] Compare the exponents
[2] Align the mantissa
[3] Add/sub the mantissa
[4] Normalize the result

$X = A \times 10^a = 0.9504 \times 10^3$
$Y = B \times 10^b = 0.8200 \times 10^2$

1) Compare exponents :
   $3 - 2 = 1$
2) Align mantissas
   $X = 0.9504 \times 10^3$
   $Y = 0.08200 \times 10^3$
3) Add mantissas
   $Z = 1.0324 \times 10^3$
4) Normalize result
   $Z = 0.10324 \times 10^4$



INSTRUCTION CYCLE

Pipeline processing can occur also in the instruction stream. An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments.

Six Phases* in an Instruction Cycle

[1] Fetch an instruction from memory
[2] Decode the instruction
[3] Calculate the effective address of the operand
[4] Fetch the operands from memory
[5] Execute the operation
[6] Store the result in the proper place

* Some instructions skip some phases
* Effective address calculation can be done in the part of the decoding phase
* Storage of the operation result into a register is done automatically in the execution phase

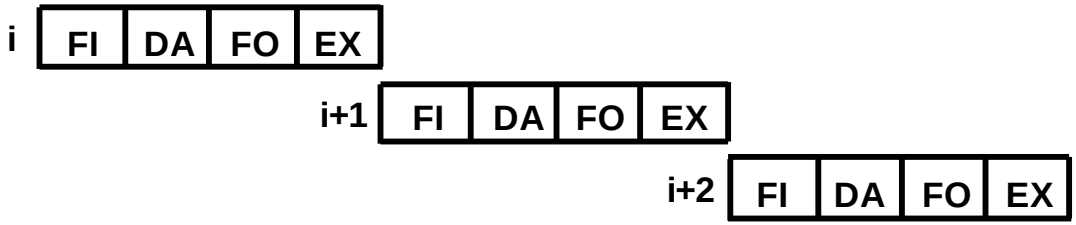
==> 4-Stage Pipeline

[1] FI: Fetch an instruction from memory
[2] DA: Decode the instruction and calculate the effective address of the operand
[3] FO: Fetch the operand
[4] EX: Execute the operation

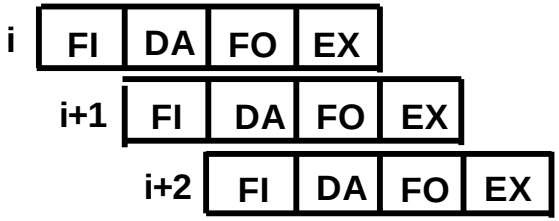
INSTRUCTION PIPELINE

Execution of Three Instructions in a 4-Stage Pipeline

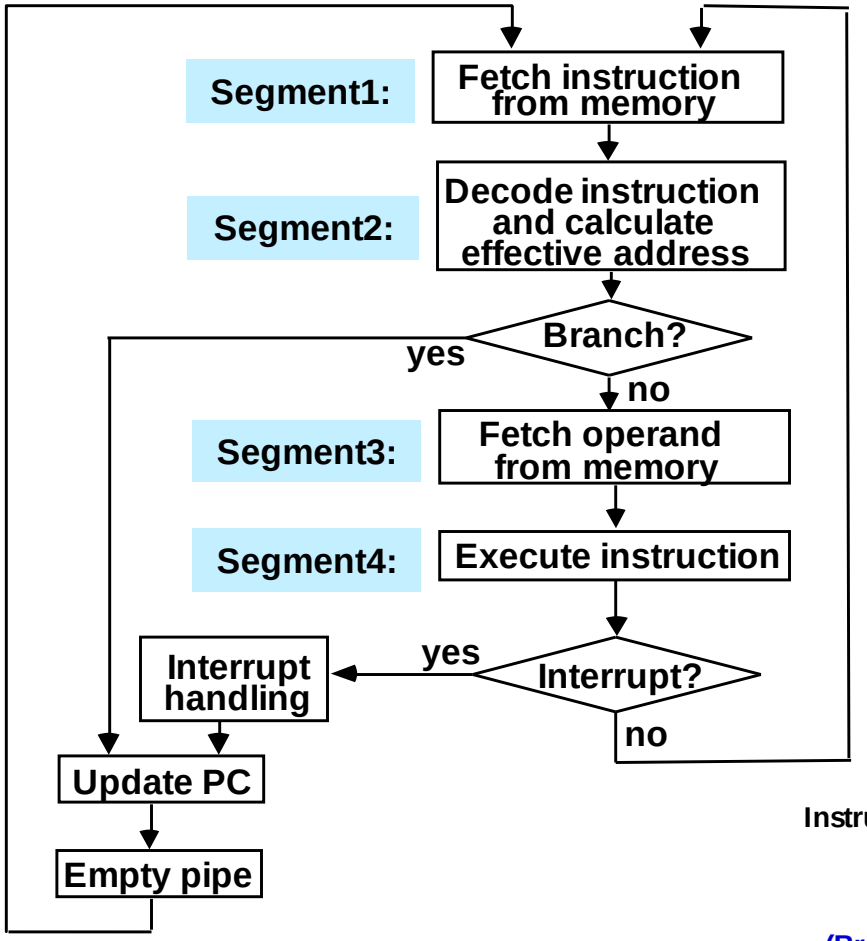
Conventional



Pipelined



INSTRUCTION EXECUTION IN A 4-STAGE PIPELINE



Instruction (Branch)

Step:	1	2	3	4	5	6	7	8	9	10	11	12	13
1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
3			FI	DA	FO	EX							
4				FI	-	-	FI	DA	FO	EX			
5					-	-	-	FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

Pipeline Conflicts

– **Pipeline Conflicts : 3 major difficulties**

1) **Resource conflicts:** memory access by two segments at the same time. Most of these conflicts can be resolved by using separate instruction and data memories.

2) **Data dependency:** when an instruction depend on the result of a previous instruction, but this result is not yet available.

   Example: an instruction with register indirect mode cannot proceed to fetch the operand if the previous instruction is loading the address into the register.

3) **Branch difficulties:** branch and other instruction (interrupt, ret, ..) that change the value of PC.

RISC Computer

- **RISC (Reduced Instruction Set Computer)**
  - Machine with a very fast clock cycle that executes at the rate of one instruction per cycle.

- **Major Characteristic**
  1. Relatively few instructions
  2. Relatively few addressing modes
  3. Memory access limited to load and store instructions
  4. All operations done within the registers of the CPU
  5. Fixed-length, easily decoded instruction format
  6. Single-cycle instruction execution
  7. Hardwired rather than microprogrammed control
  8. Relatively large number of registers in the processor unit
  9. Efficient instruction pipeline
  10. Compiler support for efficient translation of high-level language programs into machine language programs

RISC PIPELINE

- **Instruction Cycle of Three-Stage Instruction Pipeline**

  I: Instruction Fetch

  A: Decode, Read Registers, ALU Operation

  E: Transfer the output of ALU to a register, memory, or PC.

- **Types of instructions**

  - Data Manipulation Instructions
  - Load and Store Instructions
  - Program Control Instructions

VECTOR PROCESSING

- There is a class of computational problems that are beyond the capabilities of a conventional computer. These problems require a vast number of computations that will take a conventional computer days or even weeks to complete.

Vector Processing Applications

- Problems that can be efficiently formulated in terms of vectors and matrices
  - Long-range weather forecasting - Petroleum explorations
  - Seismic data analysis - Medical diagnosis
  - Aerodynamics and space flight simulations
  - Artificial intelligence and expert systems
  - Mapping the human genome
  - Image processing

Vector Processor (computer)

- Ability to process vectors, and matrices much faster than conventional computers

VECTOR PROGRAMMING

Fortran Language

```
        DO  20  I = 1, 100
20      C(I) = B(I) + A(I)
```

Conventional computer (Mahine language)

```
      Initialize I = 0
20    Read A(I)
      Read B(I)
      Store C(I) = A(I) + B(I)
      Increment I = I + 1
      If I ≤ 100 goto 20
```

Vector computer

```
C(1:100) = A(1:100) + B(1:100)
```

VECTOR PROGRAMMING

– **Vector Instruction Format :**

Operation code	Base address source 1	Base address source 2	Base address destination	Vector length

ADD A B C 100

– **Matrix Multiplication**

» **3 x 3 matrices multiplication :**

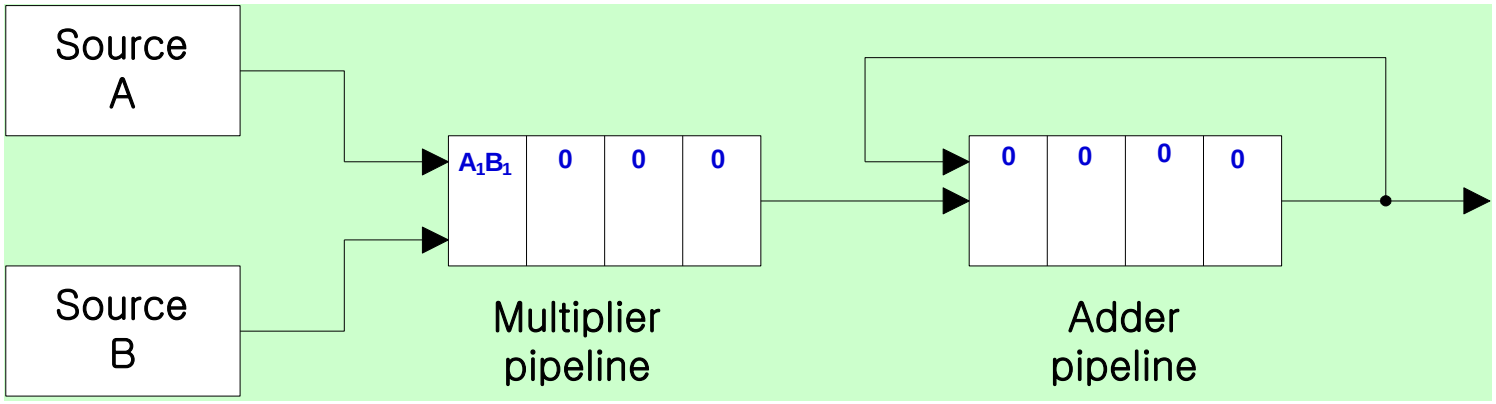
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

$$c_{11} = a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31}$$

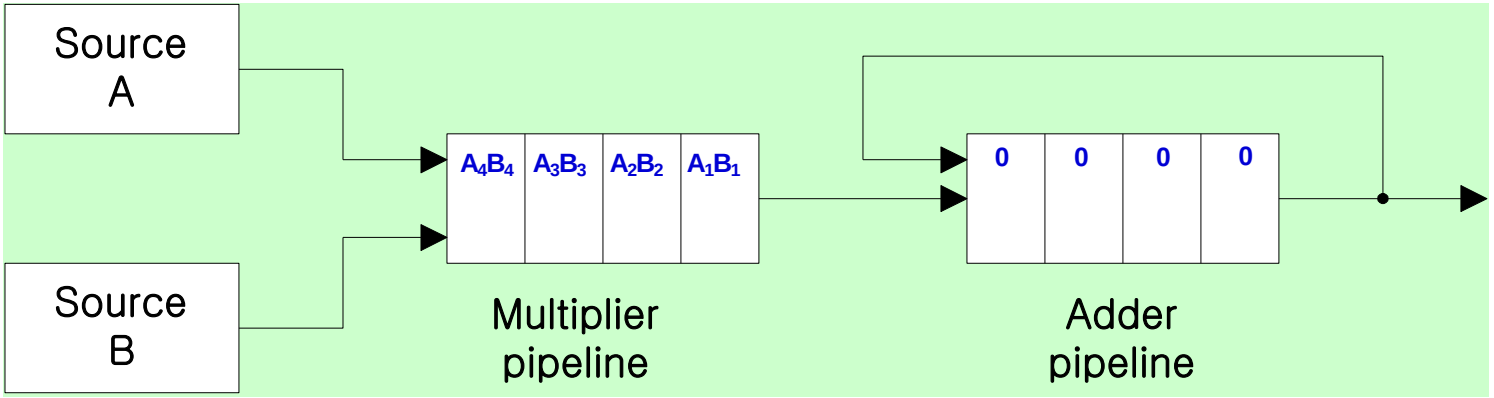
– Pipeline for calculating an inner product :
  » Floating point multiplier pipeline : 4 segments
  » Floating point adder pipeline : 4 segments

$$C = A_1B_1 + A_2B_2 + A_3B_3 + \cdots + A_kB_k$$

• after 1st clock input

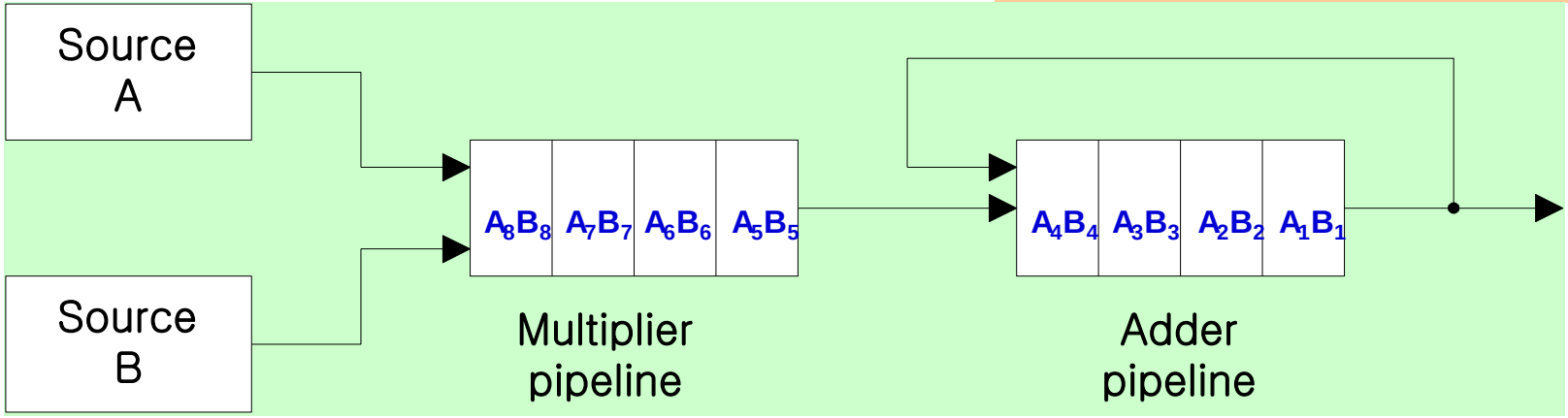


• after 4th clock input

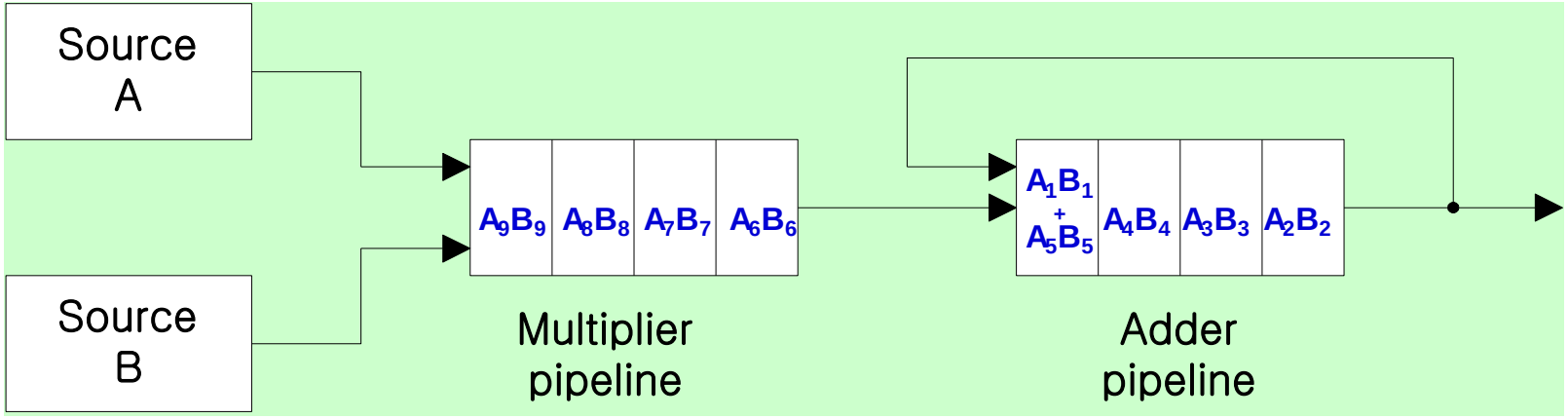


$$C = A_1B_1 + A_2B_2 + A_3B_3 + \cdots + A_kB_k$$

- after 8th clock input



- after 9th, 10th, 11th ,...



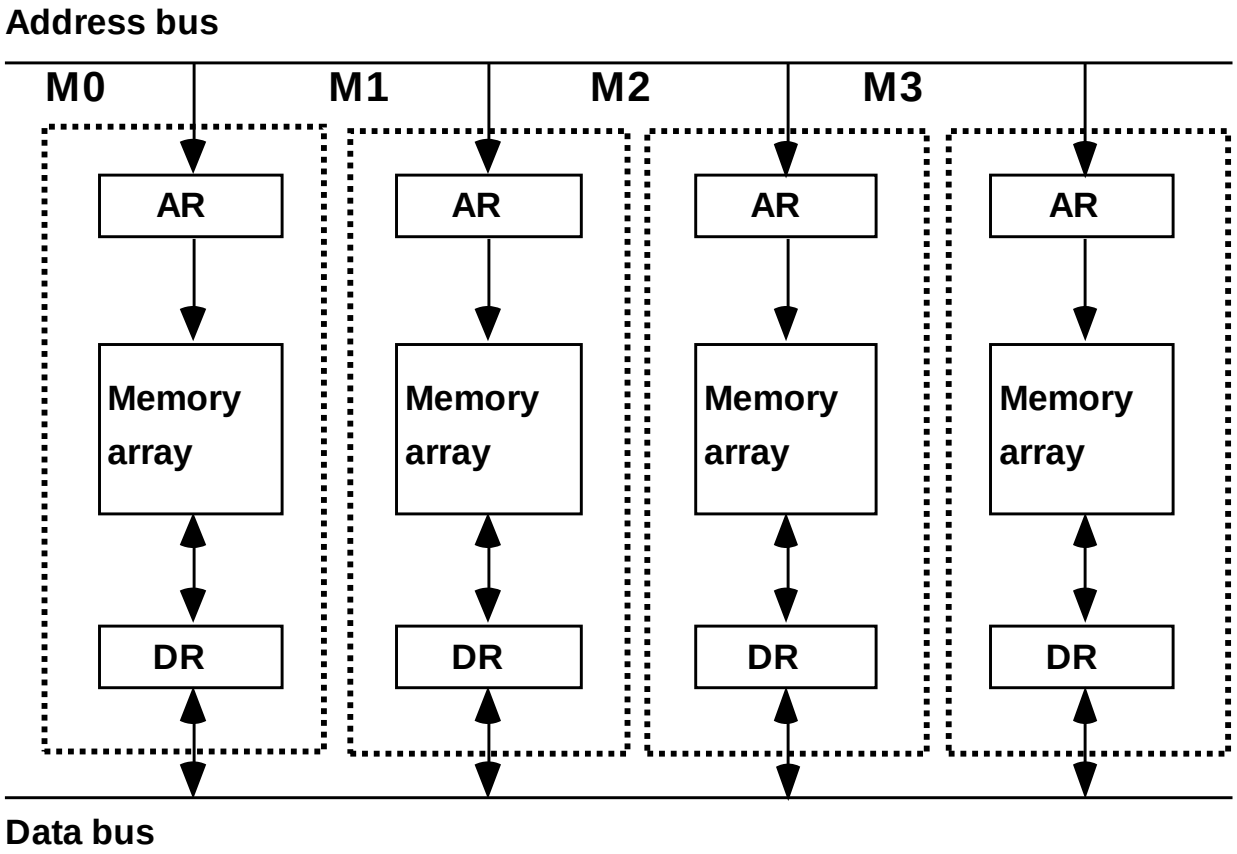
$$\begin{aligned} C = {} & A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} + \cdots \\ & + A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} + \cdots \\ & + A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} + \cdots \\ & + A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} + \cdots \end{aligned}$$

MEMORY INTERLEAVING

- Pipeline and vector processors often require simultaneous access to memory from tow or more sources.
- An instruction pipeline may require the fetching of an instruction and an operand at the same time from two different segments.
- An arithmetic pipeline usually requires two or more operands to enter the pipeline at the same time.
- Instead of using two memory buses for simultaneous access, the memory can be partitioned into a number of modules connected to common memory address and data buses.

- **Address Interleaving**
  - Different sets of addresses are assigned to different memory modules
  - For example, in a two-module memory system, the even addresses may be in one module and the odd addresses in the other.

MEMORY INTERLEAVING



- A vector processor that uses an n-way interleaved memory can fetch n operands from n different modules. By staggering the memory access, the effective memory cycle time can be reduced by a factor close to the number of modules.
- A CPU with instruction pipeline can take advantage of multiple memory modules so that each segment in the pipeline can access memory independent of memory access from other segments.

Supercomputer

- Supercomputer = Vector Instruction + Pipelined floating-point arithmetic
- High computational speed, fast and large memory system.
- Extensive use of parallel processing.
- It is equipped with multiple functional units and each unit has its own pipeline configuration.
- Optimized for the type of numerical calculations involving vectors and matrices of floating-point numbers.
- Limited in their use to a number of scientific applications:
  - *numerical weather forecasting,*
  - *seismic wave analysis,*
  - *space research.*
- They have limited use and limited market because of their high price.

Supercomputer

- Performance Evaluation Index
  - » MIPS : Million Instruction Per Second
  - » FLOPS : Floating-point Operation Per Second
    - megaflops : 10^6, gigaflops : 10^9

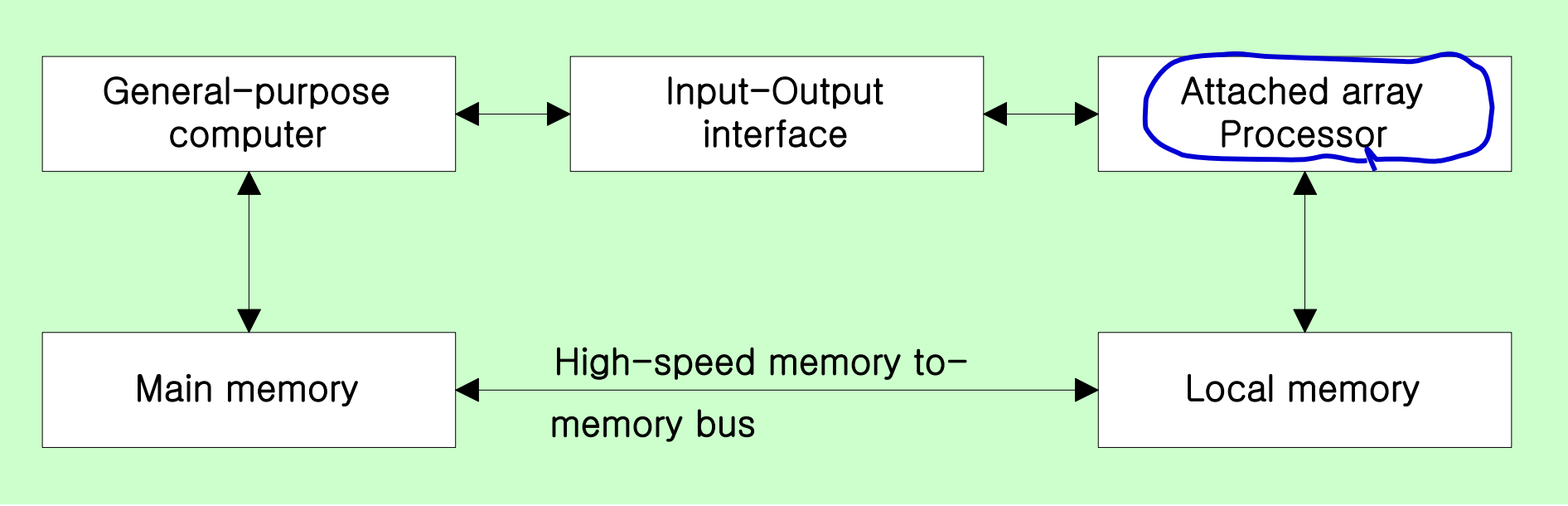
- Cray supercomputer :
  - » Cray-1 : 80 megaflops, (1976)
  - » Cray-2 : 12 times more powerful than the Cray-1
- VP supercomputer : Fujitsu
  - » VP-200 : 300 megaflops, 83 vector instruction, 195 scalar instruction
  - » VP-2600 : 5 gigaflops

9-7 Array Processors

- Performs computations on large arrays of data
  - » **Attached array processor :**
    - Auxiliary processor attached to a general purpose computer to improve the numerical computation performance.
  - » **SIMD array processor :**
    - Computer with multiple processing units operating in parallel
      - Vector C = A + B $c_i = a_i + b_i$
- Although both types manipulate vectors, their internal organization is different.

9-7 Array Processors

Attached array processor

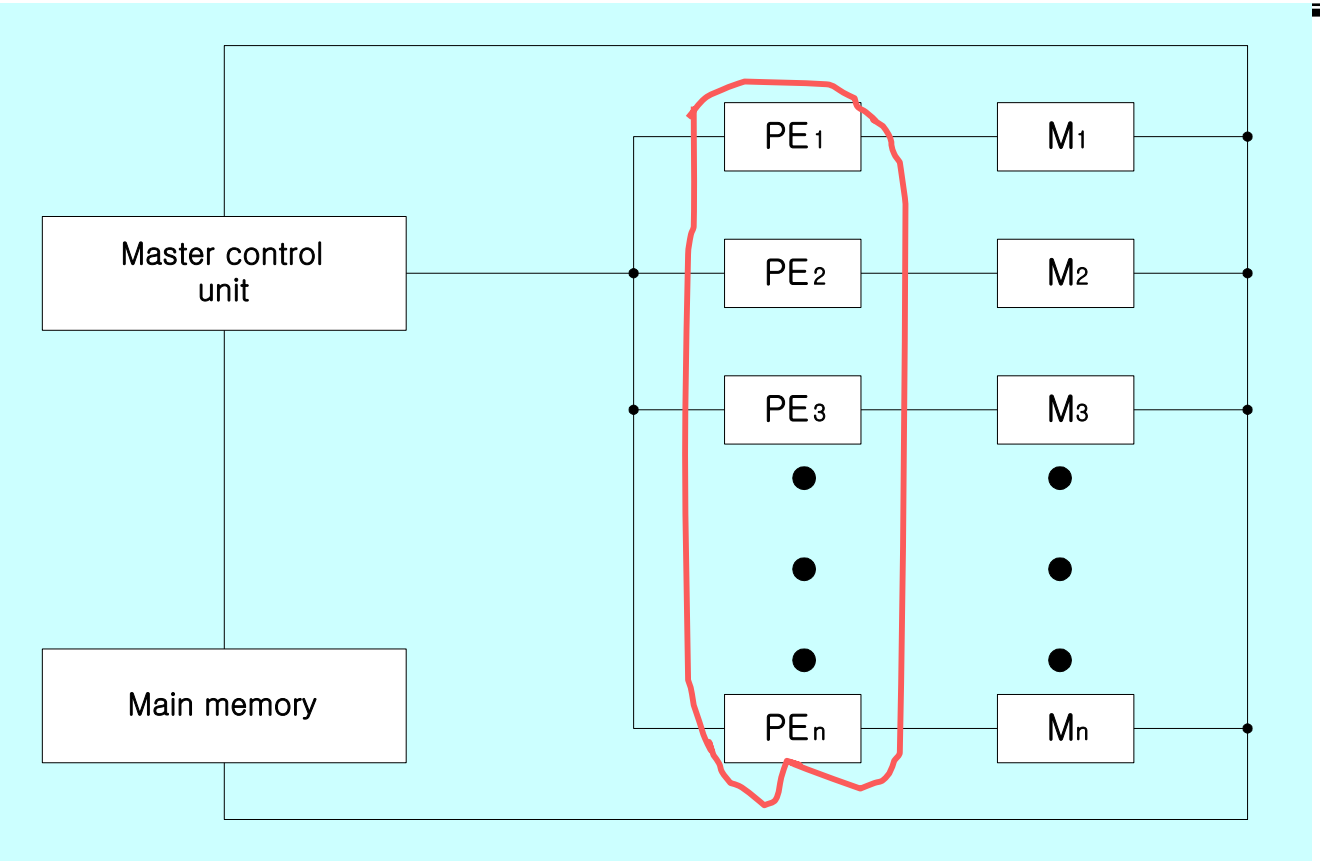


- Designed as a peripheral for complex scientific applications attached with a conventional host computer.
- The peripheral is treated like and external interface. The data are transferred from main memory to local memory through high-speed bus.
- The general-purpose computer without the attached processor serves the users that need conventional data processing.

9-7 Array Processors

SIMD array processor

- Scalar and program control instructions are directly executed within the master control unit.



- Vector instructions are broadcast to all PEs simultaneously
- **Example**: $C = A + B$
  - The master control unit first stores the i^{th} components a_i and b_i in local memory M_i for $i = 1, 2, \ldots, n$.
  - Broadcasts the floating-point add instruction $c_i = a_i + b_i$ to all PEs
  - The components of c_i are stored in fixed locations in each local memory.